2019

An international comparison of K-12 computer science education intended and enacted curricula

Katrina Falkner

Sue Sentance

Rebecca Vivian

See next page for additional authors

Authors

Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, Leonard Busuttil, Elizabeth Cole, Christine Liebe, Francesco Maiorana, Monica M. McGill, and Keith Quille

An International Comparison of K-12 Computer Science Education Intended and Enacted Curricula

Katrina Falkner
School of Computer Science
The University of Adelaide
Adelaide, South Australia, Australia
katrina.falkner@adelaide.edu.au

Sue Sentance
Raspberry Pi Foundation
Cambridge, England, UK
sue@raspberrypi.org

Rebecca Vivian
School of Computer Science
The University of Adelaide
Adelaide, South Australia, Australia
rebecca.vivian@adelaide.edu.au

Sarah Barksdale
Department of Curriculum and Instruction
University of Minnesota
Minneapolis, Minnesota, USA
barks016@umn.edu

Leonard Busuttil
Department of Technology and Entrepreneurship Education
University of Malta
Msida, Malta
leonard.busuttil@um.edu.mt

Elizabeth Cole
School of Computer Science
University of Glasgow
Glasgow, UK
e.cole.2@research.gla.ac.uk

Christine Liebe
Colorado School of Mines
Golden, Colorado, USA
cliebe@mines.edu

Francesco Maiorana
Department of Computer Science
Kansas State University
Kansas, Manhattan, USA
fmaioran@ksu.edu

Monica M. McGill
Department of Computer Science
Knox College
Galesburg, Illinois, USA
mmmcgill@knox.edu

Keith Quille
Department of Computing
TU Dublin
Tallaght, Dublin, Ireland
keith.quille@it-tallaght.ie

ABSTRACT

This paper presents an international study of K-12 Computer Science implementation across Australia, England, Ireland, Italy, Malta, Scotland and the United States. We present findings from a pilot study, comparing CS curriculum requirements (intended curriculum) captured through country reports, with what surveyed teachers (n=244) identify as enacting in their classroom (the enacted curriculum). We address the extent that teachers are implementing the intended curriculum as enacted curriculum, exploring specifically country differences in terms of programming languages and CS topics implemented. Our findings highlight the similarities and differences of intended and enacted CS curriculum within and across countries and the value of such comparisons.

CCS CONCEPTS

• **Social and professional topics → Computing education**; *K-12 education*; *Adult education.*

KEYWORDS

K-12 Computer Science Education, Teachers, Curriculum, Schools, Programming, Topics, Intended Curriculum, Enacted Curriculum

ACM Reference Format:
Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, Leonard Busuttil, Elizabeth Cole, Christine Liebe, Francesco Maiorana, Monica M. McGill, and Keith Quille. 2019. An International Comparison of K-12 Computer Science Education Intended and Enacted Curricula. In *Proceedings of Koli Calling '19: 19th Koli Calling International Conference on Computing Education Research (Koli Calling '19).* ACM, New York, NY, USA, 10 pages.

Koli Calling '19, November 21–24, 2019, Koli, Finland

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
DOI: 10.1145/3364510.3364517

1 INTRODUCTION

New primary and secondary school Computer Science (CS) curricula have recently been introduced to a number of countries (e.g. [5, 9, 18, 19, 21, 37, 48]), leading to a number of national and international efforts to develop, support and evaluate curriculum development. While K-12 CS curricula is well entrenched in some countries [4, 15, 23], it is a relatively new phenomenon in many. This poses challenges not only in the development and implementation of new curricula, but also in preparing and supporting teachers as they transition from initial teacher qualifications and experience in other learning areas to the teaching of computing [13].

There is a difference between the intended curriculum, defined by relevant standards, and the enacted or implemented curriculum, which is taught by teachers in the classroom [42]. Nolet and

McLaughlin [39] define the enacted curriculum as the operationalisation of intended curriculum, embodying the decisions teachers make in terms of what actually is taught, and how. There have been a number of initiatives aiming to identify and describe intended curriculum in a comparable, standardised way, producing country and regional reports that describe the CS curriculum standards, topics and frameworks that outline the curriculum teachers are expected to deliver [6, 25, 27, 36, 52–54]. However, with CS curriculum implementation still in its infancy across many countries, there has been little work in recording the enacted curriculum.

The enacted curriculum is defined by Porter and Smithson [42] as "actual curricular content that students engage in the classroom", and can be seen in what content is being delivered within the classroom, as well as the pedagogical approaches adopted, and - with particular relevance to CS curriculum - their use of technology, physical computing devices and tools. Descriptions of classroom practice enable us to better understand what is actually happening within our classrooms in contrast to what we intend our curriculum to be, and to identify where resources and support can be most effectively targeted and are most urgently needed to increase alignment. Careful analysis of the relationship of the intended and enacted curriculum can inform the efforts of professional development providers and pre-service teacher programs. Ultimately, the conscious monitoring of efforts from intended curriculum stakeholders and teachers enacting curriculum can facilitate robust K-12 CS education. Furthermore, there is a unique opportunity afforded through the study of enacted curriculum within the K-12 CS curriculum space. Due to its infancy, there is much that we do not understand yet about K-12 CS pedagogical practice and suitability of tools, programming languages, and physical computing. Thus, an instrument that can assist in the evaluation of the intended and enacted curriculum longitudinally and internationally will speed the development of K-12 CS pedagogy.

In this paper, we present results captured by two instruments designed to measure K-12 CS enacted and intended curriculum developed as an outcome of a 2019 ITiCSE Working Group [22] (report under review). We present results from a pilot study (n=244) across seven countries (Australia, England, Ireland, Italy, Malta, Scotland, and United States). We present our initial analysis of intended versus enacted curriculum regarding two specific aspects: programming language modality and motivation and CS topics taught, along with year level alignment in intended curriculum.

2 LITERATURE

2.1 K-12 CS Implementation

There has been a growing body of work emerging about the learning and teaching of CS in K-12 across various countries. A number of country and regional reports have been produced with the aim of identifying and describing in a comparable, standardised way, the intended curriculum, defined by Porter and Smithson [42] as "such policy tools as curriculum standards, frameworks, or guidelines that outline the curriculum teachers are expected to deliver". Gander [25] and Balanskat and Engelhardt [6] have explored K-12 CS curriculum initiatives across Europe, while several reports have been undertaken for initiatives in the UK [53, 54], the US [26], Wales [36] and Poland [52], as well as dedicated special journal issues toward case studies in K-12 CS [27].

In 2011 an ITiCSE Working Group [29] collected and validated research findings about secondary CS curricula from different countries, and in the process developed a category system (Darmstadt Model) to support comparisons across regional and national boundaries. Expanding this work, a 2015 Working Group applied the Darmstadt Model to analyse articles within two TOCE K-12 CS education special issues [27]. This work sought to understand CS curricula, goals and competencies, programming languages, tools adopted, assessment practices and teacher training, however, the authors acknowledged that the work is limited to the analysis of selected journal publications.

A Working Group in 2013 investigated trends of CS as a subject in schools by inviting CS education and teaching professionals worldwide to complete an online questionnaire about the current state of K-12 CS curriculum in their country [47]. Experts from 22 countries responded, addressing CS topics and goals covered as well as teaching methods, however, a limitation was that results were based on a small group of experts.

Prior work has set a strong foundation for understanding the state of K-12 CS curriculum and implementation efforts, however, there is an opportunity to further expand this work to focus on measuring and comparing what K-12 teachers are doing in classrooms across the world and how that compares to their intended curriculum.

2.2 Studies of CS Enacted Curriculum

van Veen et al [57] identified early in the development of CS curriculum the need for supporting both intended and enacted curriculum in curriculum descriptions. However, there has been little work within the CS domain on capturing and describing enacted modern CS curriculum, less so in a generalisable and standardised manner.

Bienkowski and Snow [10] describe initial work on a mixed-methods instrument to study curriculum enactment and teaching quality, focusing on inquiry and computational thinking practices. Rutstein et al [45] describe their evaluation instrument for curriculum implementation within the content of the ECS (Exploring Computer Science) program, aiming to characterise the relationship between measures of curriculum implementation and student learning outcome. In this work, enactment is defined purely as "lessons modified, skipped, and added", however factors that impact enactment are identified and captured within the analysis, including the learning context, as aspects of human capital, social capital and technical and resource capital. They describe the development of a series of six surveys, encompassing background and teacher professional development, with the remaining surveys addressing specific units within the ECS curriculum. Prescott et al [43] explore the experience of two middle school science teachers integrating computational thinking concepts into their science class.

Bell et al [9] describe a rich case study of secondary computing curriculum implementation in New Zealand. They discuss both the intended curriculum and standards environment, as well as a case study of enacted curriculum, analysing responses across two survey periods for an unpublished survey, with n=91 [56] and n=109 [55] respectively over the two periods. In their survey, they

gather information on teacher motivation, background demographics, implementation of standards, programming language selection, and confidence levels (including explanation of Math concepts).

Vivian and Falkner [58] conducted a survey of Australian Digital Technologies (Computer Science) teachers (n=113) to gather information on enacted curriculum, with a focus on assessment practices, reporting confidence and self-efficacy against teaching and assessing a range of CS topics. Teachers were asked to describe "any formative and summative assessment activities, processes, dialogue, instruments or resources" that they used with the context of assessing a programming activity, providing a rich description of enacted curriculum for this specific aspect of K-12 CS curriculum.

Curriculum change, either through curriculum reform or the introduction of new curriculum, poses many challenges and may take several years for full implementation to occur, particularly with alignment between enacted and intended curriculum. Broadly, teachers have identified lack of resources and time as key obstacles in implementation of new curriculum [14], in addition to the complexity of developing a clear understanding of curriculum standards [46]. There is a common confusion amongst teachers [59], but also present in government and school leadership [9, 12] regarding the distinction between Information and Communications Technology (ICT) literacy and CS, clearly indicating complexity in understanding curriculum standards. The required degree of technological awareness in CS curriculum is a further challenge, with teachers' lack of confidence and familiarity with CS tools and physical devices leading to deviation from lesson plans [35]. Black et al. [11] describe a related experience, with early adopters focusing more on fun activities, engaging with impressive technology or physical computing devices, rather than providing opportunities for deep learning of computational thinking. Additionally, the plethora of free scripted lesson plans allow CS teachers to disengage from offering intentional pedagogy. Teachers unfamiliar with technology, national curriculum or standards, may sacrifice their agency as a teacher by duly following prescribed lesson plans or modules that teach CS. Measuring the alignment between intended and enacted curriculum helps elucidate teacher confusion, while also identifying areas where greater alignment is sought, and hence resourcing needed. The impact here can be significant, with Gamoran et al. [24] and Kurz [32] identifying correlation between the alignment of intended and enacted curriculum with student performance. Gamoran et al. study this alignment in the context of transition Mathematics courses, analysing the performance of 882 students across 42 courses, identifying that curriculum alignment accounted for the majority of achievement difference between courses.

While intended curricula across the world specify a transition, either through replacement or addition, from visual through to text-based programming, there is little research as yet on this transition point, age- or developmental-appropriateness, or the potential consequences on learners in engaging with different modalities or paradigms at different points. Being able to identify what languages (and modalities or paradigms) are taught in classrooms will assist in identifying when this transition is being undertaken, if at all, helping us target future research and resource development.

In their study of programming languages and environments for novice programmers, Kelleher and Pausch [31] describe the wide range of programming languages that have been introduced to lower barriers to learning programming, introducing a range of visual programming languages and environments that aim to simplify the mechanics of programming, and provide additional learner support and motivation. While they identify that many have been influenced by common general purpose programming languages, and as such, present opportunities for translation between the two, there is a need for further research in this area.

Lewis [33] explores a case study of fifth grade students, identifying little difference in motivation, perceptions of ease of learning and use, but did identify a higher confidence of learners in their general purpose programming language treatment group. The most extensive work in this area is that undertaken by Weintrop [61, 62], exploring multiple studies within a high school setting contrasting visual, text and hybrid programming languages, identifying differences in the development of understanding of core CS concepts, and differences in student learning outcome and attitude that vary over time dependent on language modality. However, Weintrop identifies that there remain many open questions relating to how visual programming languages are perceived, and whether they aid or hinder subsequent transition to learning text-based programming languages [60].

3 RESEARCH QUESTIONS

Enacted curriculum in classrooms should reflect the curriculum policies of the state (the intended curriculum) [42]. This led us to interrogate our data to investigate the following research questions:

- What are the similarities and differences across countries in terms of intended CS curriculum topics and programming requirements?
- To what extent are teachers addressing the intended CS curriculum with their enacted curriculum in classrooms?

4 METHODS

Two instruments were developed by the 2019 ITICSE Working Group [22] for this study: *a country report template* and a *teacher survey instrument*. We briefly describe the instruments [1] as the 2019 Working Group Report (in review) presents evidence of reliability (internal consistency, and inter-rater) and validation (construct, population and sampling) using the pilot results.

4.1 Country Report Instrument

Although primarily setting out to investigate the enacted curriculum, we identified a need to capture broad curriculum information to support analysis and comparisons [2]. Information about country demographics and their intended CS curriculum were captured using a country report template. The country report template is to be completed by the survey administrator for the particular country, supported by a combination of their expert knowledge of the CS curriculum and background research for their country.

Building on a number of country and regional reports (e.g. [1, 8, 16, 28, 29, 34, 54]), we defined the following as captured in the country report template: 1) country demographics and information relating to schools (e.g. such as total population, number of schools,

[1] https://csedresearch.org/tool/?id=185

number of teachers); 2) CS curriculum state or country plan standards and requirements; 3) year level (with age for comparisons) mapped to prescribed curriculum and programming requirements, and; 4) general CS topics covered.

There was a challenge in capturing implementation of CS topics across countries, due to the differences in CS curricula and because it was dependent on whether a specific CS curriculum was available. Therefore, it was decided that a comprehensive measure of CS topics being implemented was a key consideration of the enacted curriculum and would be captured via the survey instrument. However, as a broad comparison across countries, we reviewed various curriculum analysis reports [8, 34, 48] as well as previously mentioned country reports, and from here developed a list of broad CS topics for a high-level country comparison. Future work will refine this list based on the pilot teacher survey results.

4.2 Teacher Survey Instrument

The Working Group undertook a collaborative, iterative process to develop an international teacher survey instrument based on prior work and shared expertise. The Group developed and refined a set of key survey categories that were of interest internationally in terms of enacted CS education curriculum. Collaboratively, members curated questions from surveys with evidence of reliability and validity, resulting in 88 initial example questions from 11 sources [3, 7, 17, 20, 30, 38, 40–42, 44, 51]. Additional survey items were developed based on shared group expertise and previous non-validated studies where survey items did not exist. The developed survey included 11 sections and a total of 53 questions. This paper focuses on investigating the questions relating to curriculum topics and programming languages enacted in the classroom for comparison against intended curriculum requirements.

4.3 Sample

The final dataset included 244 responses. The majority of participants (68%) were from the USA (n=115) and England (n=52). Italy, Ireland, and Scotland make up 24% of the sample, and Australia and Malta represent 8% of the participants (table 1). Some 61% of participants identified as female, 37% as male, 0.4% as other, with 1.6% preferring not to say. Over half of teachers were between the ages of 40-59 (63%) and a little less than a quarter were between the ages of 30-39 (24%). A small number (4.1%, n=10) of teachers self-identified as having a disability. The majority of teachers had been teaching for 12 or more years (49.6%, n=121), followed by 15.2% (n=37) teaching for 8-11 years. Less than 2% of teachers identified as having less than 3 years experience.

Most teachers reported working in public or government schools (84.4%, n=206; 11.9%, n=29 private schools). Some 36% (n=88) of teachers indicated that they were in a disadvantaged school (average socio-economic background of students is below the national average). A total of 39.8% of teachers indicated their school had less than 25% of low socio-economic students and 22.5% (n=55) indicated they had 50% or more students from low socio-economic backgrounds. Most of the teachers were located in urban (45.1%, n=110) or metropolitan (19.7%, n=48) regions with 29.1% (n=71) from rural or remote areas. Table 6 presents the breakdown of teachers and year levels they teach across countries.

Table 1: Participants per country

Country	N	%
Australia	14	6
England	52	21
Ireland	19	8
Italy	20	8
Malta	6	2
Scotland	18	7
USA	115	47
Total	**244**	**100**

5 INTENDED CURRICULUM

This section presents the results of the intended curriculum across countries as captured by the country reports. However, first, to contextualise the intended curriculum we present country demographics and additional information for each country.

5.1 Country Demographics

Tables 2-4 show a snapshot of the Working Group members view of their country or state intended CS curriculum. Members also provided key contextual information to expand their country/state tabulated snapshot data (see Section 5.1.1).

Table 3 shows a set of initial CS curriculum concepts and whether they are present in country/state curricula. Although starting with a smaller set of general CS concepts for capturing intended curriculum in the country report, we invited teachers in the accompanying survey to respond to their implementation of a broader set of topics (see Table 5). This was driven by our desire to capture topics that may be taught by teachers as part of their conceptualisation of CS but are not typically considered core in CS curricula. Table 3 shows that for four out of the nine countries/states with a K-6 state plan, all cover "computational thinking", "algorithms and programming" and "impact of computing". In the 7 countries/states with a state plan for students Grade 7 on-wards, curriculum content includes "computational thinking", "computer systems", "networks and internet", "data and analysis", "algorithms and programming" and "impact of computing" concepts are covered.

In our analysis, we identified that CS curricula could be categorised intro three broad types: those with a state plan for CS in place, those with no state plan for CS in place and those whose CS state plan is in development. CS guidance for those with a state plan was through standalone or embedded across disciplines. All teachers have flexibility of implementation within their state plan curricula and the opportunity to plan the delivery of lessons and resources (see Table 4). However, we do notice many variations in pre-service and in-service teacher training requirements across countries and this may impact on curriculum implementation, particularly as research has found teachers struggle with the complexity of developing a clear understanding of curriculum standards [46] and confusion amongst teachers about what CS is [59].

The choice of programming language to be used within a CS curriculum is often left up to the individual teacher, with many

curriculum standards silent on programming language, and - sometimes - modality and/or paradigm choice. This flexibility aids teachers in that they are able to identify what best suits their immediate context, however it also poses challenges in terms of development and provision of suitable resources, assessment and professional development, as well as increased expectations on teacher capability and preparedness [9]. Programming languages for those countries/states with a state plan use visual programming, or block-based, languages through K-6. From Grade 6/7 onwards, we observe text-based programming being introduced, either as a transition from, or in addition to, visual programming.

5.1.1 Additional context information. In capturing country report data, it was identified there is a need to provide additional information to expand on the data in the tables to explain some of the intricacies and to provide supporting contextual information. We include the descriptions for countries in this study below.

In *Australia* CS commences from the first year of school until year 10. No national curriculum is mandated at the final stages of secondary school (Grade 11 and Grade 12) because courses are optional for students and align to final certification. CS curriculum is at the early stages of implementation, with each state or territory determining reporting requirements. As a result, reporting expectations vary for both government and privately funded schools. Formal pre-service training and in-service professional CS learning varies in terms of requirements and availability.

England has a mandatory computing curriculum in state-funded schools from age 5-16 (Year 1-11) which covers computer science, information technology and digital literacy. This can be seen in the representation of teaching year levels of teachers from England who took the survey (see Table 6). At age 14, students can additionally elect to take a GCSE in CS, and at age 16, an A Level in CS. Postgraduate initial teacher training courses have been available, with financial incentives, for secondary computing teachers since 2013. The government supported the Network of Excellence [50] for in-service professional development of computing from 2013-2018 with a small amount of funding, and then massively increased the amount of support by forming the National Centre for Computing Education in 2018 to support in-service teachers [49].

In *Ireland* secondary school education is in two phases including the Junior Cycle at age 12-15 followed by the Leaving Certificate (which includes fifth and sixth year). These phases/years are mandatory across all schools. There is an optional year, TY (also known as transition year or fourth year). In the Junior cycle students undertake short courses across a range of subject areas which includes an optional in coding. In 2018 Ireland finalised the pilot upper secondary CS curriculum and by September 2020 all schools will be eligible to implement the CS curriculum at their own choosing. In primary, the CS curriculum is under development and is expected to be launched in 2022. The pilot phase involved a school choosing their own concepts and content which will be used to develop the curriculum. Although the secondary curriculum is optional teachers have control to decide on resources and pedagogy.

In *Italy* the secondary schools vary in specialisation, including academic, technical and vocational. CS is not mandatory in all types of high school but it is delivered in secondary schools specialising in technology or science. Object orientated programming is mandatory in the higher stages of technical schools. CS is promoted in primary and lower secondary, with CS guidance that includes "computational thinking" concepts. Formal reporting takes place in some secondary schools.

Since 2018-2019 in *Malta*, all pupils from year 7-11 follow an ICT C3 certificate which includes CS education. In the primary years Computational Thinking learning objectives are embedded in the Digital Literacy cross-curricula theme and the teacher decides how and when to implement them. These are not formally assessed. CS is a standalone subject at year 9 comprised of two branches, one being VET IT (based on networking and vocational/hands-on) and the other Computing (including programming, databases, computer architecture). Secondary schools formally report on CS in years 7-11. Pre-service CS training is compulsory for teachers delivering CS from years 7-11.

In *Scotland* all pupils have an entitlement from pre-school up to 3rd year in secondary school to a Broad General Education (BGE). Across the BGE computing science guidelines are organised into a discrete subject. However, teachers and schools have ownership on its delivery. Fourth year to 6th year computing science is optional for qualifications. In 2016 the computing science curriculum K-10 Broad General Education for curriculum content for computing science was refreshed.

In the *United States (US)* there is no national CS curriculum, however, individual states can mandate their own CS curriculum to be implemented. If there is no state or district wide curriculum formally adopted then primary and secondary schools have autonomy to implement CS curriculum and/or classes, often using the Computer Science Teachers Association (CSTA) standards as a framework. For the purposes of this paper, we have used the CSTA standards to reflect on implementation due to the variances between states. State funding is sometimes available for CS in-service professional development through various initiatives.

6 ENACTED CURRICULUM

This section presents results and discussion from the survey investigating what teachers are doing in classrooms, including CS topics and programming languages.

6.1 Computer Science Topics

Figure 1 presents an overview of the total percentage of teachers, across all countries, who indicated they are teaching particular CS topics. We observe that topics mostly being taught are algorithms, programming, computational thinking and data representation. Topics taught less are Machine Learning and Artificial Intelligence.

Examining topic implementation more closely across countries, Table 5 shows the total results for each country and the topics that teachers have identified as implementing in the classroom. Topics included within intended curriculum are highlighted with an asterisk and bold font. Although this is a pilot study, due to varied response rates per country, this preliminary data illustrates the usefulness of the instruments in helping to explore and visualise differences in intended and enacted curriculum across countries. For example, we can observe across the table, topics such as "Artificial Intelligence" and "robotics" are only included within intended

Table 2: Overall school-related demographic information for countries.

COUNTRY/USA STATE	AUSTRALIA (AUS)	COLORADO (US-CO)	ENGLAND (ENG)	IRELAND (IRL)	ITALY (ITA)	ILLINOIS (US-IL)	MALTA (MLT)	MINNESOTA (US-MN)	SCOTLAND (SCO)
Population (million)	25.09	5.69	55.62	4.70	60.50	12.7	0.47	5.6	5.44
No. of schools	9,477	1,900	29,972	3,961	8,636	4,266	170	2,066	2,400
No. of students	3,893,834	911,536	8,378,809	920,867	8,422,419	2,072,880	46,247	862,971	693,251
No. of teachers (FTE)	288,583	59,989	498,100	66,327	872,268	135,701	2,976	57,262	51,959

Table 3: CS concepts in curricula across pilot study states and countries: Explicit (✓) Implicit (❖) Not covered (×)

Concepts	AUS	US-CO	ENG	IRL	ITA	US-IL	MLT	US-MN	SCO
Computational Thinking	✓	✓	✓	✓	✓	×	✓	×	✓
Computer Systems	✓	❖	✓	✓	✓	×	❖	×	✓
Networks and Internet	✓	❖	✓	✓	✓	×	✓	×	✓
Data & Analysis	✓	✓	✓	✓	✓	×	✓	×	✓
Algorithms and Programming	❖	❖	✓	✓	✓	×	❖	×	✓
Impact of Computing	✓	✓	✓	✓	✓	×	✓	×	✓

Table 4: Demographics of pilot study countries/states education systems.
(i) Yes (✓) No (×) Additional information (❖)
(ii) Pre-service training - Varies(V) Compulsory (✓), Elective (E) *Date previous CS curriculum refreshed.

COUNTRY/USA STATE	AUS	US-CO	ENG	IRL	ITA	US-IL	MLT	US-MN	SCO
CS State or country plan	✓	×	✓	×	❖	×	✓	×	✓
CS Curriculum k-6 standards defined	✓	×	✓	❖	❖	×	✓	×	✓
CS Curriculum: Y7+ standards defined	✓	✓	✓	❖	❖	×	✓	×	✓
CS Guidelines - standalone subject	✓	✓	✓	❖	❖	×	❖	❖	✓
CS Guidelines - across disciplines	×	×	×	×		×	❖		×
Teacher autonomy to implement state/country guidelines as standalone or cross discipline	✓	✓	✓	×		✓	❖		✓
CS Formal Reporting	V	×	×*	×	❖	×	❖	❖	❖
CS in pre-service training Primary	E	E	✓	E	✓	E	×	×	E
CS in pre-service training Secondary	E	E	✓	E	E	E	✓	×	✓
CS training for in-service Primary	V	✓	E	V		×	✓	✓	
CS training for in-service Secondary	V	✓	E	V		✓	✓	✓	
Year endorsed	2015	2018	2013/14	×	×	×	2018*	×	2016*

curriculum in a single country each, despite being included in the enacted curriculum by teachers across a number of countries.

This illustrates, further, differences in alignment between intended and enacted curriculum, with multiple examples of concepts included in the intended curriculum with low rates of reporting in enacted curriculum, e.g. "hardware" across Australia, Ireland and Malta, and more starkly, "ethics" in Australia. In contrast, there are concepts that feature strongly in the reported enacted curriculum, e.g. "computational thinking" and "hardware" in England, and "robotics" in Australia that may suggest concepts that teachers find are useful or relevant to include or link to within the intended curriculum. However, discrepancies between intended and enacted curriculum may impact on student performance, as has been found in previous research [24, 32].

Table 5: Comparison of intended and enacted curriculum topics across countries. *Included in intended curriculum

CS Topics	Australia	England	Ireland	Italy	Malta	Scotland	USA
Algorithms	**79%***	**100%***	**68%***	**70%***	**33%***	**100%***	**82%***
Artificial Intelligence	7%	44%	32%	10%	0%	6%	**30%***
Computational Thinking	**57%***	96%	68%	**45%***	17%	**89%***	72%
Cybersecurity	71%	83%	16%	35%	17%	**72%***	**57%***
Data analysis and visualisation	**29%***	44%	26%	25%	0%	11%	**43%***
Data representation (e.g. digital data, binary)	**57%***	**88%***	**53%***	**45%***	**33%***	100%	**68%***
Databases	14%	71%	42%	**45%***	**17%***	89%	*27%
Design process (or Design Thinking)	**86%***	**54%***	**58%***	**20%***	17%	56%	72%
Ethics	**29%***	**88%***	58%	35%	0%	**56%***	**75%**
Hardware	**26%***	90%	**68%***	**55%***	**50%***	**94%***	**61%***
Information Systems	**50%***	58%	21%	**30%***	33%	**72%***	35%
Machine Learning	7%	23%	26%	5%	17%	11%	21%
Networks and Digital Systems	**64%***	90%	16%	40%	**17%***	**39%***	**45%***
Privacy	**64%***	77%	42%	40%	**17%***	**61%***	**64%***
Programming skills and concepts	**79%***	**100%***	**100%***	**80%***	**50%***	**100%***	**87%***
Robotics	79%	33%	42%	40%	**50%***	11%	47%
Web Systems	36%	62%	**37%***	**50%***	17%	**94%***	38%
Total sample (n)	14	52	19	19	6	18	115

Table 6: Teaching year levels by country.

Country	Pre-primary (3-5 years old)	Junior Primary (6-7 years old)	Primary (8-10 years old)	Upper Primary (11-12 years old)	Lower Secondary (13-15 years old)	Secondary (16-17 years old)	Senior Secondary (18-19 years old)
USA	5	15	17	36	76	70	56
England	5	10	12	33	38	36	27
Italy	1	2	4	2	8	14	13
Ireland		2	5	7	13	13	8
Scotland	1	1	2	9	16	15	6
Australia	3	6	9	12	3	1	
Malta		2		2	2	1	1
Total	15	38	49	101	156	150	111

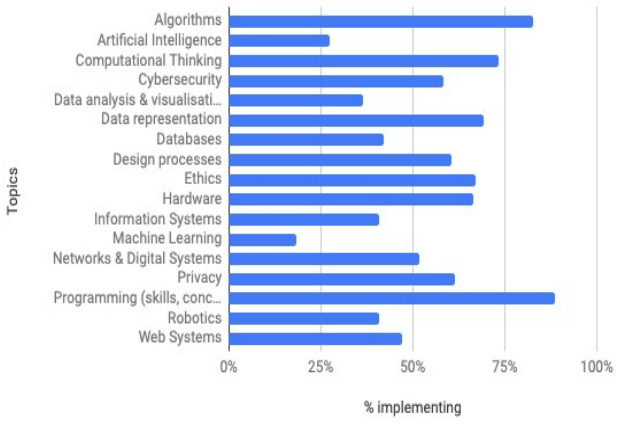

Figure 1: Teachers' implementation of CS topics across all countries combined.

6.2 Programming Languages

In Figure 2 we present the overall reported usage of the mode/paradigm of programming languages across student age groups. For this study we have investigated programming language implementation according to the following classifications:

- **Unplugged**: The use of embodied programming activities or tangible sequence cards with no computers.
- **Symbolic (no text block-based)**: Visual programming environments that utilise symbols with no text (e.g. ScratchJR).
- **Visual (text block-based)**: Visual programming environments that include text within blocks with constructed syntax and semantic (e.g. Scratch).
- **Hybrid**: Environments that include a combination of visual and text-based programming (e.g. Pencil Code).
- **Text-based**: Programming languages in which text are used, including the construction of syntax and semantics (e.g. Python).

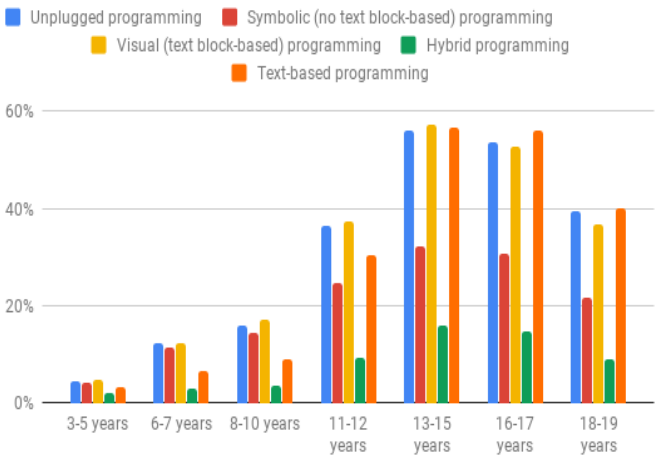

Figure 2: Programming languages implemented across age groups for all countries.

The graph demonstrates that teachers are using both unplugged and visual programming increasingly from ages 10-17, with a gradual increase in the number of teachers using text-based programming from ages 11-19. Hybrid programming also feature somewhat as being used through the transition period from visual to text-based programming (8-19 years of age).

Further breaking this down by country, we see in Figure 3 comparison across countries for programming languages implemented across age groups according to percentage of respondents from those countries. The highlighted sections indicate where curricula stipulates what type of programming language is to be used for that particular age group (as identified through the country report instrument). Table 6 shows the distribution of teaching year levels within the sample as reported by country for cross-reference. What we observe in Figure 3 is quite a number of differences between intended and enacted programming languages, and in particular the use of "unplugged programming" despite this only being explicitly mentioned in Australia's early years CS curriculum.

Table 7 shows the mean comparison across countries for agreement towards reasons for selecting programming environments. Overall, we observe that across countries there were strong consistencies between teachers' motivations for using programming languages and environments, suggesting that regardless of curriculum or location that teachers share similarities. What is interesting to note here is that teachers' motivations for using particular programming environments is driven by student-focused reasons, such as age appropriateness, stages in students' learning development and capabilities, scaffolding reasons, as well as factors relating to resource availability and cost. On the other hand, we observe that institutional factors, such as national and school determined curriculum requirements do not play a large role in teachers' decisions.

Although here teachers rated their confidence as only mildly impacting on programming choice, prior work has found that that confidence with CS tools and devices can result in deviations in teaching [35]. Further work with a larger sample is required to

Country	Ages	Un plugged	Symbolic (no text)	Visual (text)	Hybrid	Text based
Australia	3-5	21%	21%	21%	14%	7%
	6-7	43%	43%	43%	14%	14%
	8-10	64%	64%	64%	14%	29%
	11-12	79%	79%	86%	21%	43%
	13-15	14%	7%	21%	0%	14%
	16-17	7%	7%	7%	0%	7%
	18-19	0%	0%	0%	0%	0%
England	3-5	8%	6%	10%	4%	10%
	6-7	17%	15%	17%	6%	13%
	8-10	21%	17%	21%	6%	13%
	11-12	60%	40%	54%	10%	58%
	13-15	65%	37%	58%	10%	73%
	16-17	63%	31%	52%	8%	67%
	18-19	48%	21%	33%	2%	48%
Ireland	3-5	0%	0%	0%	0%	0%
	6-7	5%	4%	4%	0%	0%
	8-10	16%	6%	8%	2%	11%
	11-12	21%	8%	12%	2%	21%
	13-15	47%	10%	23%	2%	68%
	16-17	47%	12%	23%	2%	68%
	18-19	26%	10%	15%	0%	42%
Italy	3-5	0%	0%	5%	0%	0%
	6-7	5%	5%	10%	0%	0%
	8-10	15%	15%	20%	5%	10%
	11-12	10%	10%	10%	0%	10%
	13-15	30%	35%	40%	10%	40%
	16-17	45%	55%	55%	20%	65%
	18-19	40%	50%	50%	15%	60%
Malta	3-5	0%	0%	0%	0%	0%
	6-7	0%	17%	0%	0%	0%
	8-10	0%	0%	0%	0%	0%
	11-12	17%	17%	33%	17%	17%
	13-15	33%	33%	17%	33%	33%
	16-17	17%	17%	0%	17%	17%
	18-19	0%	17%	17%	0%	0%
Scotland	3-5	6%	6%	0%	0%	0%
	6-7	6%	6%	6%	0%	6%
	8-10	11%	6%	11%	0%	11%
	11-12	44%	17%	50%	11%	50%
	13-15	83%	50%	89%	17%	89%
	16-17	78%	50%	83%	17%	83%
	18-19	33%	22%	33%	11%	33%
USA	3-5	3%	3%	3%	1%	2%
	6-7	10%	8%	9%	2%	5%
	8-10	10%	9%	10%	2%	4%
	11-12	28%	16%	28%	10%	19%
	13-15	60%	31%	61%	23%	51%
	16-17	56%	27%	55%	20%	51%

Figure 3: Comparison across countries for programming languages implemented across age groups.

determine if there are any notable relationships between teacher confidence and access to PD and their implementation of CS topics, programming languages, tools and resources.

7 CONCLUSIONS

This study presents pilot results from 244 participants across seven countries. We have focused this paper on the alignment between intended and enacted curriculum in the areas of topics taught and programming languages used. We see these as critical areas for further analysis and monitoring not only in terms of alignment and its ensuing benefits, but also in relation to our assumptions as tertiary educators on prerequisite knowledge and experience.

We have identified that both visual and text-based programming languages are being used across K-12 by some teachers, warranting further research into potential impact on student learning and motivations. We also identify that unplugged activities are commonly used across K-12, extending into senior years despite this

Table 7: Mean comparison across countries for agreement towards reasons for selecting programming environments.

Reason for use	Australia	England	Ireland	Italy	Malta	Scotland	US	All
Appropriateness for age	4.1	4.2	4.1	4.2	4.0	4.1	4.2	4.1
Cost or availability	4.1	4.1	4.1	4.1	4.2	4.1	4.1	4.1
Devices available	3.8	3.8	3.8	3.8	3.8	3.8	3.8	3.8
Stage in students' learning	4.1	4.0	4.0	4.0	4.0	4.0	4.1	4.1
Scaffolding learners	4.1	4.0	4.0	4.0	4.0	4.0	4.1	4.1
School determined	2.8	2.8	2.7	2.7	2.6	2.7	2.7	2.8
Curriculum determined	3.2	3.2	3.1	3.1	3.2	3.2	3.2	3.2
Supporting resources available	3.6	3.6	3.5	3.6	3.6	3.5	3.6	3.6
My confidence level	3.7	3.8	3.7	3.7	3.7	3.7	3.7	3.7
Purpose of the activity (e.g. robotics)	3.8	3.9	3.8	3.9	3.7	3.8	3.9	3.8
What students can do (tutorial/open)	3.9	3.9	3.8	3.9	3.8	3.8	3.9	3.9

not being explicitly defined in intended curricula. Furthermore, we notice teachers' motivations for programming language choice is consistent across countries. Interestingly we expected that curriculum would drive teachers' motivations for selecting programming languages, however, our results discovered this isn't the case and that student-driven factors motivate selection.

A limitation of this study is that results are based on a small pilot sample size, particularly for some countries, however, future work will seek to survey a larger sample across multiple countries. Nevertheless, this study demonstrates the value in investigating intended versus enacted curricula in terms of recording teachers' curriculum enactment and in identifying differences with curriculum alignment across countries. These insights can potentially be used to guide further curriculum reform, or the development of targeted resources and/or professional development to better support teachers in implementing and delivering new CS curricula.

This paper has focused exclusively on the aspects of CS topics and programming languages implemented, however, there are opportunities to explore other forms of enacted curriculum such as CS resources used and pedagogy. A further limitation is that we have focused our analysis and presentation of results on K-12 broadly, with future research warranting a breakdown of analysis into primary and secondary years to determine if there are differences, as well as other factors that impact on programming language and CS topic implementation. We recommend ongoing research to continue to survey and monitor the landscape to determine whether enacted curriculum implementation changes over time as intended curriculum implementation in countries matures.

REFERENCES

[1] 2018. *2018 State of Computer Science Education.* Technical Report. https://code.org/files/2018{_}state{_}of{_}cs.pdf

[2] J Ainley and R Carstens. 2018. Teaching and Learning International Survey (TALIS) 2018 Conceptual Framework. (2018), 2018 pages. https://doi.org/10.1787/799337c2-en

[3] John Ainley and Ralph Carstens. 2018. Teaching and Learning International Survey (TALIS) 2018 Conceptual Framework. 187 (2018). https://doi.org/https://doi.org/10.1787/799337c2-en

[4] Mansour M. Al-Sulaiman. 1999. A Computing Curriculum for Technical High Schools in the Kingdom of Saudi Arabia. *J. King Saud Univ. Comput. Inf. Sci.* 11 (Jan. 1999), 85–104. https://doi.org/10.1016/S1319-1578(99)80005-6

[5] Australian Curriculum, Assessment and Reporting Authority (ACARA). 2015. Australian Curriculum: Digital Technologies. http://www.australiancurriculum.edu.au/.

[6] Anja Balanskat and Katja Engelhardt. 2014. *Computing our future Computer programming and coding - Priorities, school curricula and initiatives across Europe.* Technical Report. https://doi.org/10.1111/j.1465-7295.200

[7] A. Bandura. 2006. *Guide for Constructing Self-Efficacy Scales.* Age Information Publishing, Greenwich. 307–337 pages.

[8] Erik Barendsen, NataÅąa Grgurina, and Jos Tolboom. 2016. A new informatics curriculum for secondary education in The Netherlands. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* https://doi.org/10.1007/978-3-319-46747-4{_}9

[9] T. Bell, P. Andreae, and A. Robins. 2014. A case study of the introduction of Computer Science in NZ schools. *ACM Transactions on Computing Education* 14 (2014), 1–31.

[10] Marie Bienkowski and Eric Snow. 2017. Studying Implementation of Secondary Introductory Computer Science: Pilot Results (Abstract Only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17).* ACM, New York, NY, USA, 703–703. https://doi.org/10.1145/3017680.3022432

[11] Jonathan Black, Jo Brodie, Paul Curzon, Chrhttps://www.overleaf.com/project/5d2843014a4c0d57d252 Myketiak, Peter Mcowan, and Laura R. Meagher. 2013. Making computing interesting to school students: Teachers' perspectives. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE,* 255–260. https://doi.org/10.1145/2462476.2466519

[12] Neil C. Brown, Michael Kölling, Tom Crick, Simon Peyton Jones, Simon Humphreys, and Sue Sentance. 2013. Bringing computer science back into schools: lessons from the UK. In *44th ACM technical symposium on Computer Science education.* ACM, Denver, Colorado, USA, 269–274.

[13] Neil C. C. Brown, Sue Sentance, Tom Crick, and Simon Humphreys. 2014. Restart: The Resurgence of Computer Science in UK Schools. *ACM Trans. Comput. Educ.* 14, 2, Article 9 (June 2014), 22 pages. https://doi.org/10.1145/2602484

[14] Charalambos Y. Charalambous and George N. Philippou. 2010. Teachers' concerns and efficacy beliefs about implementing a mathematics curriculum reform: integrating two lines of inquiry. *Educational Studies in Mathematics* 75, 1 (01 Sep 2010), 1–21. https://doi.org/10.1007/s10649-010-9238-5

[15] Valentina Dagienė. 2008. Teaching Information Technology and Elements of Informatics in Lower Secondary Schools: Curricula, Didactic Provision and Implementation. In *Proceedings of the 3rd International Conference on Informatics in Secondary Schools - Evolution and Perspectives: Informatics Education - Supporting Computational Thinking (ISSEP '08).* Springer-Verlag, Berlin, Heidelberg, 293–304. https://doi.org/10.1007/978-3-540-69924-8_27

[16] Valentina Dagiene, T Jevsikova, Carsten Schulte, Sue Sentance, and N Thota. 2013. A comparison of current trends within Computer Science teaching in school in Germany and the UK. In *International Conference on Informatics in Schools (ISSEP),* Ira Diethelm (Ed.). Oldenburg, Germany, 63–75.

[17] Joy-Anne D'Anca. 2017. *Mindset and Resilience: An Analysis and Intervention for School Administrators.* Ph.D. Dissertation. St. John's University (New York), School of Education and Human Services.

[18] Department for Education. 2013. *The National Curriculum in England.* Department for Education Government of UK, Crown, Cheshire.

[19] Directorate for Learning and Assessment Programmes. 2019. SEC Syllabus (2019): Computing. https://www.um.edu.mt/__data/assets/pdf_file/0017/292310/SEC09.pdf.

[20] Carol S. Dweck. 2008. *Mindset: The new psychology of success.* Random House Digital, Inc.

[21] Education Scotland. 2017. Benchmark Technologies. education.gov.scot/improvement/documents/technologiesbenchmarkspdf.pdf.

[22] Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, Leonard Busuttil, Elizabeth Cole, Christine Liebe, Francesco Maiorana, Monica M. McGill, and Keith Quille. 2019. An International Benchmark Study of K-12 Computer Science Education in Schools. In *Proceedings of the 2019 ACM Conference on Innovation*

and Technology in Computer Science Education (ITiCSE '19). ACM, New York, NY, USA, 257–258. https://doi.org/10.1145/3304221.3325535

[23] Judith Gal-Ezer, Catriel Beeri, David Harel, and Amiram Yehudai. 1995. A High School Program in Computer Science. *Computer* 28, 10 (Oct. 1995), 73–80. https://doi.org/10.1109/2.467599

[24] Adam Gamoran, Andrew C. Porter, John Smithson, and Paula A. White. 1997. Upgrading High School Mathematics Instruction: Improving Learning Opportunities for Low-Achieving, Low-Income Youth. *Educational Evaluation and Policy Analysis* 19, 4 (1997), 325–338. https://doi.org/10.3102/01623737019004325 arXiv:https://doi.org/10.3102/01623737019004325

[25] W Gander, A Petit, G Berry, B Demo, J Vahrenhold, A McGettrick, R Boyle, M Drechsler, A Mendelson, C Stephenson, C Ghezzi, and B Meyer. 2013. *Informatics Education: Europe Cannot Afford to Miss the Boat.* Technical Report. Association for Computing Machinery &, Joint Informatics Europe ACM Europe Working Group on Informatics Education, New York. 1–21 pages.

[26] Hai Hong, Jennifer Wang, and Sepehr Hejazi Moghadam. 2016. K-12 computer science education across the U.S.. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-46747-4{_}12

[27] Peter Hubwieser, Michal Armoni, and Michail N. Giannakos. 2015. How to Implement Rigorous Computer Science Education in K-12 Schools? Some Answers and Many Questions. *ACM Transactions on Computing Education* (2015). https://doi.org/10.1145/2729983

[28] Peter Hubwieser, Michail N. Giannakos, Marc Berges, Torsten Brinda, Ira Diethelm, Johannes Magenheim, Yogendra Pal, Jana Jackova, and Egle Jasute. 2015. A Global Snapshot of Computer Science Education in K-12 Schools. In *ITiCSE Working Group Reports*. ACM, Vilnius, Lithuani, 65–83. https://doi.org/10.1145/2858796.2858799

[29] Peter Hubwieser, Sigrid Schubert, Michal Armoni, Torsten Brinda, Valentina Dagiene, Ira Diethelm, Michail N. Giannakos, Maria Knobelsdorf, Johannes Magenheim, and Roland Mittermeir. 2011. Computer science/informatics in secondary education. In *Proceedings of the 16th annual conference reports on Innovation and technology in computer science education - working group reports - ITiCSE-WGR '11*. https://doi.org/10.1145/2078856.2078859

[30] Ilkka Jormanainen. 2018. On Computer Science Major Students' Motivation in a Practically Oriented Robotics Course. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling '18)*. ACM, New York, NY, USA, Article 29, 2 pages. https://doi.org/10.1145/3279720.3279749

[31] Caitlin Kelleher and Randy Pausch. 2005. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Comput. Surv.* 37, 2 (June 2005), 83–137. https://doi.org/10.1145/1089733.1089734

[32] Alexander Kurz, Stephen Elliott, Joseph H. Wehby, and John Smithson. 2010. Alignment of the Intended, Planned, and Enacted Curriculum in General and Special Education and Its Relation to Student Achievement. *The Journal of Special Education* 44 (11 2010), 131–145. https://doi.org/10.1177/0022466909341196

[33] Colleen M. Lewis. 2010. How Programming Environment Shapes Perception, Learning and Goals: Logo vs. Scratch. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, USA, 346–350. https://doi.org/10.1145/1734263.1734383

[34] Linda Mannila, Valentina Dagiene, Barbara Demo, Natasa Grgurina, Claudio Mirolo, Lennart Rolandsson, and Amber Settle. 2014. Computational Thinking in K-9 Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference - ITiCSE-WGR '14*. https://doi.org/10.1145/2713609.2713610

[35] Orni Meerbaum-Salant, Michal Armoni, and Mordechai (Moti) Ben-Ari. 2010. Learning Computer Science Concepts with Scratch. In *Proceedings of the Sixth International Workshop on Computing Education Research (ICER '10)*. ACM, New York, NY, USA, 69–76. https://doi.org/10.1145/1839594.1839607

[36] F. Moller and T. Crick. 2016. A National Engagement Model for Developing Computer Science Education in Wales. In *The 9th International Conference on Informatics in Schools,*. Munster, Germany, 1–13.

[37] National Council for Curriculum and Assessment. 2018. Computer Science Curriculum Specification. https://www.curriculumonline.ie/Senior-cycle/Senior-Cycle-Subjects/Computer-Science.

[38] National Network of Education Research Practice Partnerships. 2019. Research-Practice Partnerships (RPP) for CS common data collection 9-12. Personal communication.

[39] Victo Nolet and Margaret J. McLaughlin. 2000. *Accessing the general curriculum: Including students with disabilities in standards-based reform.* Thousand Oaks, CA: Corwin Press, Inc.

[40] Outlier Research & Evaluation. 2017. BASICS Study ECS Teacher Implementation and Contextual Factor Questionnaire Measures [Measurement scales]. http://outlier.uchicago.edu/basics/resources/MeasuresTeacherImplementation/

[41] Paul R. Pintrich, David A. F. Smith, Teresa Garcia, and Wilbert J. McKeachie. 1991. A Manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ). *Mediterranean Journal of Social Sciences* 6, 1 (1991), 156—164.

[42] Andrew C. Porter and John L. Smithson. 2001. Defining, Developing and Using Curriculum Indicators. CPRE Research Reports, 12-2001.

[43] Paige Prescott, Irene A. Lee, and Kersti Tyson. 2019. Teacher Beliefs in Student Capabilities As a Mediating Factor in a Novel Understanding of Enactment of CT Curriculum. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. ACM, New York, NY, USA, 1277–1277. https://doi.org/10.1145/3287324.3293841

[44] Keith Quille and Susan Bergin. 2019. CS1 : how will they do ? How can we help ? A decade of research and practice research and practice. *Computer Science Education* 29 (2019), 254–282. https://doi.org/10.1080/08993408.2019.1612679

[45] Daisy W. Rutstein, Yuning Xu, Kevin McElhaney, and Marie Bienkowski. 2019. Developing Implementation Measures for K-12 Computer Science Curriculum Materials. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. ACM, New York, NY, USA, 321–327. https://doi.org/10.1145/3287324.3287424

[46] Alan Schoenfield. 2011. *How We Think*. New York: Routledge. https://doi.org/10.4324/9780203843000

[47] Carsten Schulte, Malte Hornung, Sue Sentance, Valentina Dagiene, Tatjana Jevsikova, Neena Thota, Anna Eckerdal, and Anne-Kathrin Peters. 2012. Computer science at school/CS teacher education - Koli working-group report on CS at school. In *International Conference on Computing Education Research, Koli Calling 2012*. https://doi.org/10.1145/2401796.2401800

[48] Deborah Seehorn, Stephen Carey, Brian Fuschetto, Irene Lee, Daniel Moix, Dianne O'Grady-Cunniff, Barbara Boucher Owens, Chris Stephenson, and Anita Verno. 2011. *CSTA K-12 Computer Science Standards, Computer Science Teachers Association.* ACM, New York.

[49] Sue Sentance. 2019. Moving to mainstream: developing computing for all. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education (WiPSCE '19)*. ACM. https://doi.org/10.1145/3361721.3362117

[50] Sue Sentance, Simon Humphreys, and Mark Dorling. 2014. The network of teaching excellence in computer science and master teachers. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*. ACM, 80–88.

[51] Robert H. Stupnisky, Allison BrckaLorenz, Bridget Yuhas, and Frédéric Guay. 2018. Faculty members' motivation for teaching and best practices: Testing a model based on self-determination theory across institution types. *Contemporary Educational Psychology* 53 (2018), 15–26.

[52] Maciej M. Sysło and Anna Beata Kwiatkowska. 2015. Introducing a new computer science curriculum for all school levels in Poland. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-25396-1{_}13

[53] The Royal Society. 2012. *Shut down or restart? The way forward for computing in UK schools.* Technical Report. London. https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

[54] The Royal Society. 2017. *After the reboot: Computing education in UK schools.* Technical Report. The Royal Society, London, United Kingdom. 1–116 pages. royalsociety.org/computing-education

[55] David Thompson and Tim Bell. 2013. Adoption of New Computer Science High School Standards by New Zealand Teachers. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education (WiPSE '13)*. ACM, New York, NY, USA, 87–90. https://doi.org/10.1145/2532748.2532759

[56] David Thompson, Tim Bell, Peter Andreae, and Anthony Robins. 2013. The Role of Teachers in Implementing Curriculum Changes. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 245–250. https://doi.org/10.1145/2445196.2445272

[57] Maarten van Veen, Fred Mulder, and Karel Lemmen. 2004. What is Lacking in Curriculum Schemes for Computing/Informatics?. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '04)*. ACM, New York, NY, USA, 186–190. https://doi.org/10.1145/1007996.1008046

[58] Rebecca Vivian and Katrina Falkner. 2018. A Survey of Australian Teachers' Self-efficacy and Assessment Approaches for the K-12 Digital Technologies Curriculum. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE '18)*. ACM, New York, NY, USA, Article 5, 10 pages. https://doi.org/10.1145/3265757.3265762

[59] Rebecca Vivian, Katrina Falkner, and Nickolas Falkner. 2014. Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. *Research in Learning Technology* 22 (Aug. 2014). https://doi.org/10.3402/rlt.v22.24691

[60] David Weintrop. 2019. Block-based Programming in Computer Science Education. *Commun. ACM* 62, 8 (July 2019), 22–25. https://doi.org/10.1145/3341221

[61] David Weintrop, Alexandria K. Hansen, Danielle B. Harlow, and Diana Franklin. 2018. Starting from Scratch: Outcomes of Early Computer Science Learning Experiences and Implications for What Comes Next. In *Proceedings of the 2018 ACM Conference on International Computing Education Research (ICER '18)*. ACM, New York, NY, USA, 142–150. https://doi.org/10.1145/3230977.3230988

[62] David Weintrop and Uri Wilensky. 2017. Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Trans. Comput. Educ.* 18, 1, Article 3 (Oct. 2017), 25 pages. https://doi.org/10.1145/3089799